TECHNICAL REPORT 95-02

Controlling Priority Inversion using Abortions

Hiroaki Takada and Ken Sakamura

January 11, 1995

DEPARTMENT OF INFORMATION SCIENCE
FACULTY OF SCIENCE, UNIVERSITY OF TOKYO

7–3–1 HONGO, BUNKYO-KU TOKYO, 113 JAPAN

TITLE

Controlling Priority Inversion using Abortions

AUTHORS

Hiroaki Takada and Ken Sakamura

KEY WORDS AND PHRASES

real-time system, scheduling algorithm, synchronization protocol, abortion, priority inversion

ABSTRACT

Making critical sections abortable is a promising approach to controlling priority inversions. To improve the schedulability of a system using abortions, the maximum number of abortions should be decreased. In this paper, we propose a real-time synchronization protocol named the *selective abort protocol (SAP)* which introduces a new abort scheme to the priority ceiling protocol. Our proposed scheme can make the maximum number of abortions smaller than with the conventional priority-based abort schemes. A method to determine an upper bound on the number of abortions under the SAP is presented, and schedulability analyses of systems using the protocol are illustrated. Some extensions of the SAP are also discussed.

REPORT DATE January 11, 1995	**WRITTEN LANGUAGE** English
TOTAL NO. OF PAGES 11	**NO. OF REFERENCES** 10

ANY OTHER IDENTIFYING INFORMATION OF THIS REPORT

DISTRIBUTION STATEMENT

First issue 45 copies. This technical report is available via anonymous FTP from `ftp.is.s.u-tokyo.ac.jp` (directory `/pub/tech-reports`).

SUPPLEMENTARY NOTES

DEPARTMENT OF INFORMATION SCIENCE
Faculty of Science, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan

Controlling Priority Inversion using Abortions

Hiroaki Takada and Ken Sakamura

Department of Information Science,
Faculty of Science, University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113, Japan

Technical Report 95-02

January 11, 1995

Abstract

Making critical sections abortable is a promising approach to controlling priority inversions. To improve the schedulability of a system using abortions, the maximum number of abortions should be decreased. In this paper, we propose a real-time synchronization protocol named the *selective abort protocol (SAP)* which introduces a new abort scheme to the priority ceiling protocol. Our proposed scheme can make the maximum number of abortions smaller than with the conventional priority-based abort schemes. A method to determine an upper bound on the number of abortions under the SAP is presented, and schedulability analyses of systems using the protocol are illustrated. Some extensions of the SAP are also discussed.

1 Introduction

Priority inversion, the state in which a higher priority task is blocked by lower priority tasks due to resource sharing, is a major cause to degrade schedulability in hard real-time systems. Some real-time synchronization protocols are proposed with which the maximum blocking duration is limited to at most the duration of one critical section of a lower priority task [1, 2]. However, in some application systems in which the maximum processing times of some critical sections are very long, the schedulability is severely degraded due to priority inversions even though one of these protocols is adopted.

For such application systems, making critical sections abortable is a promising approach to reducing priority inversions. Some real-time synchronization protocols using abortions have been proposed and evaluated [3, 4]. Though the schedulability of higher priority tasks is improved using these protocols, the schedulability of lower priority tasks is degraded because aborted critical sections must be re-executed from the beginning. In hard real-time environments in which lower priority tasks also have severe timing constraints, this re-execution time should be minimized. It is also important to evaluate the worst-case re-execution time to guarantee their schedulability. However, it is only lately that these issues are laid importance [5, 6, 7].

In this paper, we propose a real-time synchronization protocol named the *selective abort protocol* (SAP) which introduces an improved abort scheme called the *selective abort scheme* to the priority ceiling protocol (PCP) [1]. With this scheme, an *abort task set* is defined for each critical section, and only the tasks included in the abort task set can abort the critical section. The other tasks cannot abort it and are executed according to the base synchronization protocol, i.e. the PCP. Our proposed scheme can make the maximum number of abortions smaller than the conventional priority-based abort schemes [6, 7] and can improve the schedulability of a system.

For analyzing the schedulability of a system using the SAP, we present a method to determine an upper bound on the number of abortions under the protocol. Schedulability analyses of example task sets using the SAP are illustrated, and the effectiveness of the protocol is demonstrated. We also present some simplifications and extensions of the SAP.

2 Assumptions and Notations

In this paper, the same assumptions with the PCP are adopted except that critical sections are abortable. In this section, we formalize abortable critical sections after reviewing the basic assumptions adopted with the PCP. The definition of the schedulable laxity, which is a useful concept in discussing schedulability, is also presented.

2.1 Basic Assumptions

A system consists of n periodic tasks $\tau_1, \tau_2, \cdots, \tau_n$. Each task has a *priority* determined by the rate monotonic scheduling algorithm. T_i and P_i denote the period and priority of task τ_i respectively, and $\tau_1, \tau_2, \cdots, \tau_n$ are sorted in descending order of priority. Hence, $T_i \leq T_j$ and $P_i \geq P_j$ if $i \leq j$.

An execution of a task is called a *job*. τ_i also denotes a job requested by task τ_i. Each task requests a job at the beginning of each period, and the job's deadline is at the end of the period. The priority of a job is set to the priority of the requesting task when it starts, and is changed during its execution. The priority of a job at any given moment is called its *current priority* or simply its *priority*.

Each job is preemptive and executed according to the priority-driven scheduling. The highest priority job that is ready to run is executed first. Jobs with the same priority are scheduled in a FCFS order. When the execution of a job is delayed due to the execution of other jobs of higher or equal priority tasks, the job is said to be *preempted*.

Each task includes critical sections guarded by binary semaphores. The j-th critical section in task τ_i is represented as $z_{i,j}$. When a job is preempted within a critical section, the critical section is said to be preempted. Two critical sections included in a task must be either disjoint or properly nested. Job τ_i is called *blocked*, when τ_i is forced to wait for the execution of lower priority tasks.

The maximum processing time of job τ_i is denoted as C_i and the maximum blocking duration during its execution is denoted as B_i. The execution of a job is not delayed due to reasons other than being preempted or blocked. When the schedulability of a system is discussed, we ignore the overheads of task scheduling, context switches, and other processing needed for task synchronization.

2.2 Abortable Critical Sections

An abortable critical section consists of an abortable segment followed by an unabortable segment. When a job is executed within the abortable segment, the execution of the critical section may be aborted and restarted from the beginning. Once the job enters the unabortable segment, the critical section is not aborted and is executed to the end.

The maximum processing times of the abortable segment of $z_{i,j}$ and its unabortable segment are denoted as $A_{i,j}$ and $U_{i,j}$ respectively. Though the maximum processing time of the entire critical section is less than or equal to $A_{i,j} + U_{i,j}$ in general, we assume that it is equal to $A_{i,j} + U_{i,j}$ in this paper. Also, we ignore various overheads when aborting a critical section for simplicity. In the following, we refer to an abortable critical section as defined above simply as a critical section. A critical section $z_{i,j}$ is called unabortable when $A_{i,j} = 0$. We use the word *abortable* to make $A_{i,j} \neq 0$ explicit.

If abortable critical sections are nested arbitrarily, evaluation of the maximum processing time of the outer critical section becomes complicated. Thus, when $z_{i,j}$ and $z_{i,j'}$ are nested, we allow only the following two cases ($z_{i,j'}$ is assumed to be the outer critical section, here).

- $z_{i,j}$ is unabortable and is included in the unabortable segment of $z_{i,j'}$.
- The abortable segment of $z_{i,j}$ is identical with that of $z_{i,j'}$, and the unabortable segment of $z_{i,j}$ is included in that of $z_{i,j'}$. When one of the critical sections is aborted, the other is regarded to also be aborted.

	C_i	T_i	P_i	$A_{i,1}+U_{i,1}$	B_i	L_i
τ_1	4	10	P_1	-	0	6
τ_2	4	15	P_2	2	4	-1
τ_3	4	30	P_3	2	4	2
τ_4	10	100	P_4	4	0	8

Table 1: Task Set for Example 1

	C_i	T_i	P_i	$A_{i,1}+U_{i,1}$	B_i	L_i
τ_1	4	10	P_1	-	0	6
τ_2	3	15	P_2	2	4	0
τ_3	4	20	P_3	2	4	-2
τ_4	10	100	P_4	4	0	9

Table 2: Task Set for Example 2

As a result of these simplifications, when $z_{i,j}$ is aborted m times during its execution, its maximum processing time is prolonged by $mA_{i,j}$. The maximum duration that $z_{i,j}$ can block a higher priority job that can abort $z_{i,j}$ is reduced to $U_{i,j}$.

2.3 Schedulable Laxity

The schedulable laxity L_i of task τ_i is the time duration having the following properties. Even if the maximum blocking duration of τ_i is prolonged by a duration shorter than or equal to L_i, the task is schedulable. If it is prolonged by a duration longer than L_i, the task becomes unschedulable. When τ_i is not schedulable, L_i takes a negative value having the property that τ_i becomes schedulable if the maximum blocking duration of τ_i is shortened by a duration longer than or equal to $-L_i$.

Under the PCP, the schedulable laxity of each task can be derived by transforming the formula of the necessary and sufficient condition of the schedulability [8] as the following.

$$L_i = \max_{(k,l)\in\mathcal{R}_i} \left[lT_k - \sum_{r\in\mathcal{P}_i} C_r \left\lceil \frac{lT_k}{T_r} \right\rceil \right] - B_i,$$

where $\mathcal{P}_i = \{\ j \mid j = 1,\cdots,n,\ and\ P_j \geq P_i\ \}$ and $\mathcal{R}_i = \{(k,l) \mid k \in \mathcal{P}_i,\ l = 1,\cdots,\lfloor T_i/T_k \rfloor\}$.

3 Necessity of Abortions

In this section, we present some example task sets which cannot be scheduled with the PCP because the execution times of some critical sections are quite long. These task sets can be scheduled with our proposed protocol, which will be shown later in Section 5.

Example 1 The first example task set consisting of four tasks is presented in Table 1. Here, τ_2, τ_3, and τ_4 include critical sections $z_{2,1}$, $z_{3,1}$, and $z_{4,1}$ respectively. These critical sections are guarded by the same binary semaphore. The columns of B_i and L_i present the maximum blocking duration and the schedulable laxity of each task respectively when the task set is scheduled with the PCP.

From this table, τ_2 is found to be unschedulable because L_2 is negative. To make τ_2 schedulable, its maximum blocking duration must be reduced. By making $z_{4,1}$ abortable and permitting τ_2 to abort it, τ_2 becomes schedulable. Because the schedulable laxity of τ_2 is -1, $z_{4,1}$ should be divided into the abortable and the unabortable segments such that $A_{4,1} = 1$ and $U_{4,1} = 3$. In order to minimize the number of abortions, only τ_2 should be permitted to abort $z_{4,1}$ and the other jobs should not abort it.

Example 2 The second example task set is presented in Table 2. τ_2, τ_3, and τ_4 include critical sections $z_{2,1}$, $z_{3,1}$, and $z_{4,1}$ respectively guarded by the same binary semaphore. The columns of B_i and L_i have the same meanings with Table 1.

In this example, τ_3 is not schedulable with the PCP. To make τ_3 schedulable, τ_3 must be permitted to abort $z_{4,1}$. Because the schedulable laxity of τ_3 is -2, $z_{4,1}$ should be divided such that $A_{4,1} = 2$ and $U_{4,1} = 2$. In order to minimize the number of abortions, the jobs other than τ_3 should not abort $z_{4,1}$.

4 Selective Abort Protocol

4.1 Definition of the SAP

Our definition of the priority ceiling is the same with the PCP, i.e. the priority ceiling of a semaphore is equal to the priority of the highest priority task that may lock the semaphore. The priority ceiling of a critical section $z_{i,j}$ is defined to be the priority ceiling of the semaphore guarding $z_{i,j}$.

A set of tasks called an abort task set is defined for each abortable critical section. $\mathcal{Z}_{i,j}$ denotes the abort task set of $z_{i,j}$.

Definition 1 (Abort Task Set) The abort task set $\mathcal{Z}_{i,j}$ of a critical section $z_{i,j}$ is defined statically such that the following conditions are satisfied.

1. The priority of a task included in $\mathcal{Z}_{i,j}$ is higher than P_i and is lower than or equal to the priority ceiling of $z_{i,j}$.
2. When two abortable critical sections are nested and their abortable segments are identical, their abort task sets must be the same. □

With the selective abort scheme, a critical section $z_{i,j}$ is aborted in its abortable segment when a job of a task included in $\mathcal{Z}_{i,j}$ is blocked or preempted. We introduce this scheme to the PCP and define the selective abort protocol (SAP) as presented below.

Definition 2 (Selective Abort Protocol) When a job τ_i tries to enter a critical section $z_{i,j}$, it is checked if any critical sections are preempted with higher priority ceilings than the priority of τ_i. If some such critical sections are preempted[1], τ_i must execute one of the following steps.

1. When these critical sections are preempted within their abortable segment and a task included in their abort task sets[2] is preempted or blocked, all of the critical sections are aborted.
2. Otherwise, the job τ_k that is executing these critical sections inherits the priority of τ_i and is executed first. When τ_k finishes the execution of the critical sections or the critical sections are aborted by another job, τ_k releases the semaphore and its priority is reset to the priority before the inheritance. Then, the execution of τ_i resumes.

Now, τ_i can lock the semaphore guarding $z_{i,j}$ and begins to execute $z_{i,j}$.

When $z_{i,j}$ is aborted during the execution of its abortable segment, the semaphore guarding $z_{i,j}$ is released and the priority of τ_i is reset to the priority it had before entering $z_{i,j}$. Then, τ_i re-executes $z_{i,j}$ from the beginning.

When a job of a task τ_k is requested and some of the critical sections whose abort task sets include τ_k are being executed or preempted within their abortable segments, all of the critical sections are aborted[3]. □

When all critical sections are unabortable, Step 1 in the above protocol definition is never executed, and the behavior of the SAP corresponds to that of the PCP. In this sense, the SAP is an extension of the PCP.

[1]It is easily proved from the definition of this protocol that all of these critical sections are included in a job.

[2]Because these critical sections are included in a job and are nested, their abort task sets are the same.

[3]It is also proved that all of these critical sections are included in a job.

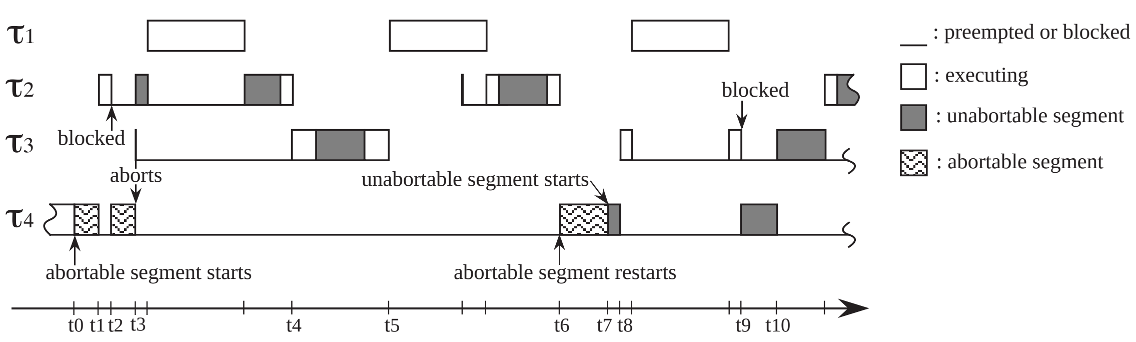

Figure 1: Scheduling Example under the SAP

4.2 Properties of the SAP

The SAP inherits the important properties of the PCP [8] presented in the following two theorems.

Theorem 3 The SAP prevents deadlocks. □

Theorem 4 A job can be blocked for at most the duration of one execution of the critical sections or their unabortable segments that can possibly block the job. □

In other words, each job is blocked no more than once. Therefore, B_k can be determined by calculating the maximum processing time of the critical sections or their unabortable segments that can possibly block τ_k. τ_k can be blocked by a critical section $z_{i,j}$ included in a lower priority task or its unabortable segment, if the priority ceiling of $z_{i,j}$ is higher than or equal to the priority of τ_k. τ_k can be blocked by the whole $z_{i,j}$, if τ_k is not included in $\mathcal{Z}_{i,j}$. Otherwise, it can be blocked only by its unabortable segment.

The following theorem is also important in evaluating the maximum overheads for aborting critical sections.

Theorem 5 During an execution of a job, at most one critical section of a lower priority task is aborted. □

In the SAP, the priority of a job is changed according to the priority inheritance policy. Another protocol adopting the stack resource policy (SRP) [2] instead is possible and has the same properties as the SAP presented above. It is also possible to apply the SRP to the unabortable segment only.

4.3 Scheduling Example under the SAP

Example 3 (Scheduling Example under the SAP) We present a scheduling example of the task set described in Example 2 using the SAP. As mentioned in Section 3, $z_{4,1}$ should be made abortable such that $A_{4,1} = 2$ and $U_{4,1} = 2$, and should be aborted only by τ_3, then $\mathcal{Z}_{4,1} = \{\tau_3\}$. Note that the priority ceiling of $z_{2,1}$, $z_{3,1}$, and $z_{4,1}$ is P_2.

Figure 1 illustrates an example scheduling of the tasks. We will describe important events in the figure.

- At time t_0, τ_4 enters the abortable segment of $z_{4,1}$.
- At time t_1, τ_2 is requested and begins to execute preempting τ_4.
- At time t_2, τ_2 tries to enter $z_{2,1}$. Though $z_{4,1}$ is preempted in its abortable segment, $z_{4,1}$ is not aborted because no task included in $\mathcal{Z}_{4,1}$ is being preempted or blocked. Therefore, τ_2 is blocked by $z_{4,1}$. τ_4 inherits the priority of τ_2 and continues to execute the abortable segment.

- At time t_3, τ_3 is requested, just before τ_4 finishes the execution of the abortable segment. Because τ_3 is included in $\mathcal{Z}_{4,1}$ and because the abortable segment of $z_{4,1}$ is being executed, $z_{4,1}$ is aborted and the priority of τ_4 is reset to P_4. Now, τ_2 resumes execution and can enter $z_{2,1}$.
- At time t_4, the execution of τ_2 is finished and τ_3 begins to execute. Because $z_{4,1}$ has been aborted, τ_3 can be executed to the end without blocking.
- At time t_6, τ_4 resumes execution. As $z_{4,1}$ has been aborted, τ_4 restarts the abortable segment of $z_{4,1}$ from the beginning.
- At time t_7, τ_4 enters the unabortable segment of $z_{4,1}$.
- At time t_8, τ_3 is requested. Because τ_4 is executing the unabortable segment of $z_{4,1}$, $z_{4,1}$ is not aborted. τ_3 begins to execute preempting τ_4.
- At time t_9, τ_3 tries to enter $z_{3,1}$. Because $z_{4,1}$ is preempted in its unabortable segment, $z_{4,1}$ is not aborted and τ_3 is blocked by $z_{4,1}$. τ_4 inherits the priority of τ_3 and resumes to execute the unabortable segment of $z_{4,1}$.
- At time t_{10}, τ_4 finishes the execution of $z_{4,1}$. The priority of τ_4 is reset to P_4.

In this example, every job meets its deadline.

5 Schedulability Analysis of the SAP

5.1 Schedulability Analysis Method

To analyze the schedulability of a system using the SAP, we can apply the results of the PCP presented below [8]. $\mathcal{P}_i$ in the following theorems denotes the index set of the tasks whose priority is higher than or equal to P_i, i.e. $\mathcal{P}_i = \{ \, j \mid j = 1, \cdots, n, \ \mathit{and} \ P_j \geq P_i \, \}$.

Theorem 6 A set of n periodic tasks using the PCP is schedulable by the rate monotonic scheduling algorithm, if

$$\forall i, 1 \leq i \leq n, \quad \sum_{r \in \mathcal{P}_i} \frac{C_r}{T_r} + \frac{B_i}{T_i} \leq i\,(2^{1/i} - 1).$$

□

Theorem 7 A set of n periodic tasks using the PCP is schedulable by the rate monotonic scheduling algorithm for all task phasings, if and only if

$$\forall i, 1 \leq i \leq n, \quad L_i = \max_{(k,l) \in \mathcal{R}_i} \left[lT_k - \sum_{r \in \mathcal{P}_i} C_r \left\lceil \frac{lT_k}{T_r} \right\rceil \right] - B_i \geq 0,$$

where $\mathcal{R}_i = \{(k,l) \mid k \in \mathcal{P}_i, \ l = 1, \cdots, \lfloor T_i/T_k \rfloor\}$. □

The latter theorem presents the necessary and sufficient condition, when C_i is equal to the maximum processing time of τ_i and B_i is equal to its maximum blocking duration. When larger values are used, this theorem provides a sufficient condition for the schedulability.

Next, we examine the effects of abortions on each parameter that appears in the theorems. By making a critical section $z_{i,j}$ abortable, the effects on the schedulability of τ_k can be examined for each of the following four cases. The maximum number of times that $z_{i,j}$ is aborted is represented as $m_{i,j}$ below.

- When P_k is higher than the priority ceiling of $z_{i,j}$, making $z_{i,j}$ abortable has no effect on any parameter of τ_k.

- When $P_k > P_i$, P_k is lower than or equal to the priority ceiling of $z_{i,j}$, and τ_k is included in $\mathcal{Z}_{i,j}$, the maximum duration that τ_k can be blocked by $z_{i,j}$ is reduced to $U_{i,j}$. Consequently, B_k can possibly be decreased by making $z_{i,j}$ abortable.
- When $P_k > P_i$, P_k is lower than or equal to the priority ceiling of $z_{i,j}$, and τ_k is not included in $\mathcal{Z}_{i,j}$, making $z_{i,j}$ abortable has no effect on any parameter of τ_k.
- When $P_k \leq P_i$, the schedulability of τ_k may be affected by the increase of C_i. C_i is prolonged by $m_{i,j}A_{i,j}$ because one abortion of $z_{i,j}$ requires additional processing time for $A_{i,j}$. B_k remains unchanged.

From these examinations, if $m_{i,j}$ can be determined for each i and j, the schedulability of the system using the SAP by the rate monotonic scheduling algorithm can be checked using Theorem 6 or 7.

The following theorem presents a method to determine an upper bound of $m_{i,j}$ when the SAP is used.

Theorem 8 $m_{i,j}$, the maximum number of times that $z_{i,j}$ is aborted when a set of n periodic tasks using the SAP is scheduled by the rate monotonic scheduling algorithm, is less than or equal to m that satisfies the following inequality.

$$\max_{(k,l)\in\mathcal{R}_{i,j,m}} \left[lT_k - \sum_{r\in\mathcal{Q}_i} C_r \left\lceil \frac{lT_k}{T_r} \right\rceil \right] \geq (m+1)A_{i,j}, \tag{1}$$

where $\mathcal{Q}_i = \{\, r \mid r = 1, \cdots, n, \ and \ P_r > P_i \,\}$
and $\mathcal{R}_{i,j,m} = \{(k,l) \mid k \in \mathcal{Q}_i, \ l = 0, 1, \cdots, \lfloor T_i/T_k \rfloor, \ and \ \sum_{\tau_r \in \mathcal{Z}_{i,j}} \lceil lT_k/T_r \rceil \leq m\}$. □

The notion of worst-case phasing [9] is used to prove this theorem. The worst-case phasing for a task is the system status in which the time until the termination of its job is the longest possible. Under static priority assignments, the worst-case phasing for a task τ_i has been proved to be the instant when every task having a higher or equal priority with τ_i requests a job [10]. This instant is called a critical instant, and is determined independently of the processing time of τ_i. From this property, the execution of the abortable segment of $z_{i,j}$ finishes latest, when a critical instant occurs immediately after τ_i enters the abortable segment of $z_{i,j}$.

Another necessary property to prove this theorem is that τ_i is not blocked within $z_{i,j}$ once the execution of $z_{i,j}$ begins. This property holds under the SAP as well as under the PCP.

As a result, the left side of (1) represents the minimum processing time given to $z_{i,j}$, while the jobs that can possibly abort $z_{i,j}$ are requested in m times or less. Here, the effect of the jobs that have equal priorities with τ_i is not counted. This is because the execution of τ_i has already started at the critical instant and precedes these jobs by the FCFS discipline. The right side of (1) represents the maximum processing time of the abortable segment, when $z_{i,j}$ is aborted m times. Therefore, if (1) is met, the execution of the abortable segment will be finished despite the abortions.

Because the left side of (1) stops increasing when $m \geq \sum_{\tau_r \in \mathcal{Z}_{i,j}} \lceil T_i/T_r \rceil$, the minimum value of m satisfying (1), if any, can be obtained by checking (1) for each m such that $1 \leq m \leq \sum_{\tau_r \in \mathcal{Z}_{i,j}} \lceil T_i/T_r \rceil$.

The upper bound of $m_{i,j}$ calculated with this method is a pessimistic bound. Accordingly, using the parameters calculated from this upper bound, Theorem 7 gives a sufficient condition for the schedulability.

5.2 Schedulability Analysis Examples

Using the method described above, we illustrate schedulability analyses of the task sets of Example 1 and 2 under the SAP.

m	LS of (1)	RS of (1)
1	0	2
2	6	3
3	6	4
4	12	5
5	12	6
6	18	7
7	18	8

Table 3: Determining an Upper Bound of $m_{4,1}$ (Eg. 1)

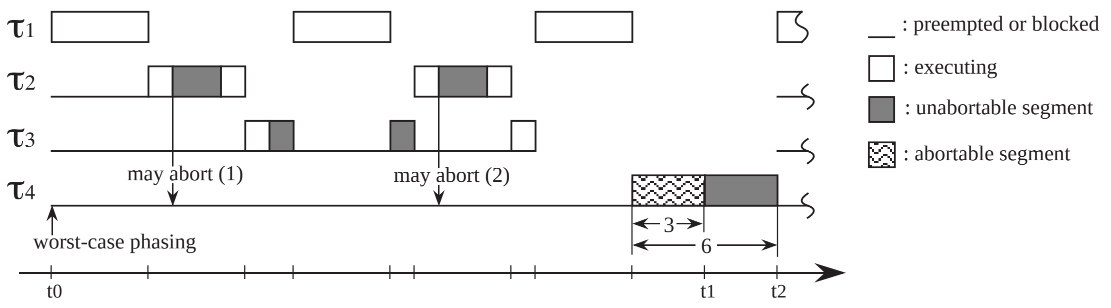

Figure 2: Determining an Upper Bound of $m_{4,1}$ (Eg. 1)

Example 4 (Schedulability Analysis under the SAP (Eg. 1)) We illustrate a schedulability analysis of the task set of Example 1 under the SAP. As mentioned in Section 3, $z_{4,1}$ should be made abortable such that $A_{4,1} = 1$ and $U_{4,1} = 3$, and should be aborted only by τ_2, then $\mathcal{Z}_{4,1} = \{\tau_2\}$.

At first, we calculate an upper bound of $m_{4,1}$ using Theorem 8. Table 3 presents the value of the left and right sides of (1) for each m such that $1 \leq m \leq \sum_{\tau_r \in \mathcal{Z}_{4,1}} \lceil T_4/T_r \rceil\ (= 7)$. From this table, we can see that (1) is first met when $m = 2$. Figure 2 illustrates the scheduling of each task, when τ_1, τ_2 and τ_3 are requested at time t_0, which is the worst-case phasing for τ_4. Until time t_1, $z_{4,1}$ can be aborted no more than twice, while the processing time given to τ_4 is sufficient to finish the execution of the abortable segment despite the abortions.

The result of calculating the effects on the schedulability of each task is presented in Table 4. In this table, C_i^+ denotes the maximum re-execution time required for τ_i when critical sections are made abortable. Since $m_{4,1}$ is known to be two or less, an upper bound of C_4^+ is $2A_{4,1}(= 2)$. Because no schedulable laxity is negative in this table, this task set can be scheduled with the SAP.

Example 5 (Schedulability Analysis under the SAP (Eg. 2)) Next, we illustrate a schedulability analysis of the task set of Example 2. In this task set, only τ_3 can abort $z_{4,1}$, i.e. $\mathcal{Z}_{4,1} = \{\tau_3\}$. Table 5 presents the value of the left and right sides of (1) for each m such that $1 \leq m \leq \sum_{\tau_r \in \mathcal{Z}_{4,1}} \lceil T_4/T_r \rceil\ (= 5)$. From this table, we can see that (1) is first met when $m = 2$. Therefore, $m_{4,1}$ is shown to be two or less by Theorem 8.

The result of calculating the effects on the schedulability of each task is presented in Table 6. Since $m_{4,1}$ is two or less, an upper bound of C_4^+ is $2A_{4,1}(= 4)$. Because no schedulable laxity is negative in this table, this task set can be scheduled with the SAP.

	C_i	T_i	P_i	$A_{i,1}$	$U_{i,1}$	B_i	C_i^+	L_i
τ_1	4	10	P_1	-	-	0	0	6
τ_2	4	15	P_2	0	2	3	0	0
τ_3	4	30	P_3	0	2	4	0	2
τ_4	10	100	P_4	1	3	0	2	6

Table 4: Schedulability Analysis under the SAP (Eg. 1)

m	LS of (1)	RS of (1)
1	2	4
2	7	6
3	12	8
4	14	10
5	19	12

Table 5: Determining an Upper Bound of $m_{4,1}$ (Eg. 2)

	C_i	T_i	P_i	$A_{i,1}$	$U_{i,1}$	B_i	C_i^+	L_i
τ_1	4	10	P_1	-	-	0	0	6
τ_2	3	15	P_2	0	2	4	0	0
τ_3	4	20	P_3	0	2	2	0	0
τ_4	10	100	P_4	2	2	0	4	5

Table 6: Schedulability Analysis under the SAP (Eg. 2)

To demonstrate the validity of the SAP, we present the results of schedulability analyses of this task set under conventional approaches. In Example 2, we have already shown that the task set cannot be scheduled with the PCP. The center columns of Table 7 show the result when $z_{4,1}$ is aborted by any higher priority task, i.e. the priority abort scheme [3] is used. In this case, $z_{4,1}$ is aborted not only by τ_3, but also by τ_2. As a result, an upper bound of $m_{4,1}$ cannot be determined with Theorem 8 and the schedulability of the task set cannot be shown. Permitting only τ_2 to abort $z_{4,1}$, which can be realized with a simpler abort scheme called the ceiling abort scheme [6], does not solve the problem. The right columns of Table 7 show this case.

5.3 Finding a Feasible Assignment of Abort Task Sets

There remains a problem how to determine the abort task set of each abortable critical section to make a given set of tasks schedulable. An assignment of abort task sets is called *feasible*, if the task set can be shown to be schedulable under the assignment with our schedulability analysis method presented above. In the SAP, a feasible assignment of abort task sets can be obtained, if any, using the following algorithm.

1. First, the null set is assigned to each abort task set.
2. The schedulable laxity of each task is calculated. If all of them are zero or larger, the current assignment of abort task sets is feasible.
3. Let τ_i be the highest priority task whose schedulable laxity is negative. τ_i is added to each of the abort task sets of the critical sections that make L_i negative. The maximum processing times of their unabortable segments must be shorter than $B_i + L_i$. If it is impossible (when $B_i + L_i < 0$, for example), feasible assignments of abort task sets do not exist.
4. As a result of the previous step, if the schedulable laxity of τ_i remains negative, feasible assignments of abort task sets do not exist.
5. Return to Step 2.

	C_i	T_i	P_i	$A_{i,1}$	$U_{i,1}$	B_i	C_i^+	L_i	$A_{i,1}$	$U_{i,1}$	B_i	C_i^+	L_i
τ_1	4	10	P_1	-	-	0	0	6	-	-	0	0	6
τ_2	4	15	P_2	0	2	2	0	2	0	2	2	0	2
τ_3	4	30	P_3	0	2	2	0	0	0	2	4	0	-2
τ_4	10	100	P_4	0	4	0	-	< 0	2	2	0	8	1

Table 7: Schedulability Analyses under the PAP and the CAP

Note that the above algorithm depends on our schedulability analysis method. If some other method is used for analyzing the schedulability, this algorithm must be modified also.

6 Simplifications and Extensions of the SAP

A priority-based abort scheme is obtained by limiting the definition of abort task sets such that $\mathcal{Z}_{i,j}$ is determined by one priority value $P^Z_{i,j}$ as follows[4]. $\mathcal{Z}_{i,j}$ includes all the tasks that have higher priority than $P^Z_{i,j}$ and have lower or equal priority with the priority ceiling of $z_{i,j}$.

With this simplification, only one value is necessary to be maintained for each critical section instead of a set of tasks. Therefore, the implementation overhead can be reduced. The abort ceiling protocol [7] is essentially the same protocol with the SRP-variant of the SAP with this simplification.

It is also possible to determine $\mathcal{Z}_{i,j}$ by more than one priority values.

The SAP can be extended in the following ways.

- The abortable segment of a critical section is divided into sub-segments, and an abort task set is defined for each sub-segment such that the tasks included in the abort task set will not increase as the execution proceeds.

  With this extension, the maximum re-execution time of the abortable segment, which is included in the right side of (1), can be reduced.

- A whole critical section is executed at a higher priority level than the priority of the task. When a critical section is aborted, its priority is reset to that priority level instead of the task's priority.

  With this extension, the execution of the critical section precedes some higher priority jobs and thus the chance of abortions is decreased. The maximum blocking durations of preceded jobs are prolonged instead.

- Different priority ceilings are assigned to the abortable and the unabortable segments of a critical section $z_{i,j}$. Though the priority ceiling of the unabortable segment must be the priority ceiling of the semaphore guarding $z_{i,j}$, that of the abortable segment can be an arbitrary value $P^A_{i,j}$ satisfying the following two conditions.

  1. $P^A_{i,j}$ must be lower than or equal to the priority ceiling of the unabortable segment.
  2. $P^A_{i,j}$ must be higher than the priority of the tasks that have lower or equal priority with the priority ceiling of the unabortable segment and that are not included in $\mathcal{Z}_{i,j}$.

  With this extension, Theorem 3 and 4, which are the most important properties of the SAP, are preserved. However, the property of Theorem 5 is lost.

  This extension can avoid some unnecessary abortions and thus the maximum number of abortions can be reduced. The ceiling abort protocol [6] is a simplified protocol of this extension.

7 Conclusion

In this paper, a real-time synchronization protocol called the selective abort protocol (SAP) is proposed and a sufficient condition for the schedulability under the protocol is presented. As an example of the SAP's applicability, we present a set of tasks which cannot be scheduled using conventional protocols, but can be scheduled using the SAP. We also discuss some simplifications and extensions of the SAP.

In this paper, the rate monotonic scheduling is adopted as the base scheduling algorithm in which the deadline of each job is at the end of each period. When the deadline of one of the jobs

[4] $P^Z_{i,j}$ must be higher than or equal to P_i and must be lower than or equal to the priority ceiling of $z_{i,j}$.

is earlier due to a jitter requirement or for some other reason, making critical sections abortable is even more effective [5, 7]. Another useful application of abortable critical sections is to incorporate them into the earliest deadline first scheduling algorithm, in which a higher priority task has a smaller schedulable laxity. Combining the selective abort scheme with the earliest deadline first scheduling remains as future work.

References

[1] R. Rajkumar, L. Sha, and J. P. Lehoczky, "Real-time synchronization protocols for multiprocessors," in *Proc. Real-Time Systems Symposium*, pp. 259–269, Dec. 1988.

[2] T. P. Baker, "Stack-based resource allocation policy for realtime processes," in *Proc. Real-Time Systems Symposium*, pp. 191–200, Dec. 1990.

[3] J. Huang, J. A. Stankovic, K. Ramamritham, and D. Towsley, "On using priority inheritance in real-time databases," in *Proc. Real-Time Systems Symposium*, pp. 210–221, Dec. 1991.

[4] H. Tokuda and T. Nakajima, "Evaluation of real-time synchronization in real-time Mach," in *Proc. USENIX Mach Symposium*, pp. 213–221, Nov. 1991.

[5] L. Shu and M. Young, "A mixed locking/abort protocol for hard real-time systems," in *Proc. Real-Time Operating Systems and Software*, pp. 102–106, May 1994.

[6] H. Takada and K. Sakamura, "Real-time synchronization protocols with abortable critical sections," in *Proc. Real-Time Computing Systems and Applications*, pp. 48–52, Dec. 1994.

[7] L. Shu, M. Young, and R. Rajkumar, "An abort ceiling protocol for controlling priority inversion," in *Proc. Real-Time Computing Systems and Applications*, pp. 202–206, Dec. 1994.

[8] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Trans. Computers*, vol. 39, pp. 1175–1185, Sept. 1990.

[9] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," in *Proc. Real-Time Systems Symposium*, pp. 166–171, Dec. 1989.

[10] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *JACM*, vol. 20, no. 1, pp. 46–61, 1973.